Chapter 1

Efficient Strategyproof Multicast in Selfish Wireless Networks

In this chapter, we study how to perform routing when each wireless node is selfish, i.e., a wireless node will always maximize its own benefit. Traditionally, it is assumed by the majority of the routing protocols for wireless networks that each wireless node will forward the packets for other nodes if it is asked to do so. However, this assumption may not be true in practice, especially when the wireless devices are owned by individual users. A node will deviate from a routing protocol if it will gain more benefit by doing so. In this chapter, we assume that each wireless node will incur a cost when it forwards a unit of data for some other nodes. A node will forward the data only if it gets a payment to compensate its cost. Its profit (or called utility) will then be the payment minus its cost if it did forward the data. For a multicast with a source node and a set of receiver nodes, we assume that they will pay the relay nodes to carry the traffic from the source to receivers. We assume that the cost of each agent is private and each agent can manipulate its reported cost to maximize its utility. A payment scheme is strategyproof if every agent maximizes its utility when it reports its cost truthfully. In this chapter, we propose several strategyproof mechanisms for multicast for selfish wireless networks when each node has a cost of forwarding a unit data based on various structures. We prove that each of our payment schemes is optimum for the corresponding structure used.

1.1 Introduction

Recent years saw a great amount of researches in wireless ad hoc networks on various important problems such as routing, Quality of Service, security, power management, and traffic and mobility modelling. However, there are still many challenges left. In wireless ad hoc networks, each host contributes its local resources to forward the data for other nodes to serve the common good, and may benefit from resources contributed by other hosts in return. Based on such a fundamental design philosophy, wireless networks provide appealing features of enhanced system robustness, high service availability and scalability. However, the critical observation that users are generally selfish and non-cooperative may severely undermine the expected wireless structure. For example, for a routing algorithm based on

the least cost path (LCP), the individual wireless node may declare an arbitrarily high cost for forwarding a data packet to other nodes since wireless nodes are energy-constraint and it is often not in the interest of a node to always relay the messages for other nodes. The root cause of the problem is, obviously, that there exist no incentives for users to be altruistic. Following the common belief in neoclassic economics, it is more reasonable to assume that all wireless terminals are *rational*: they try to maximize their benefits instead of conforming to the existing protocols. Thus, we need to design some mechanisms to assure that these *rational* wireless terminals will conform to our protocols without any deviation.

How to achieve cooperation among wireless terminals in network was previously addressed in [3, 11, 13, 2, 4, 16, 17]. The key idea behind these approaches is that terminals providing a service should be remunerated, while terminals receiving a service should be charged. Both of these methods belong to so called *credit based method*. Some of these algorithms need some special hardware which is not very practical in the real world. In recent years, *incentive based methods* have been proposed to solve the non-cooperative problem. The most well-known and widely used incentive based methods is so called the family of VCG mechanisms by Vickrey [18], Clarke [5], and Groves [9]. Nisan and Ronen [14] provided a mechanism belonging to VCG family to assure the cooperation for unicast problem in a general network where each communication link is assumed to be selfish and rational.

While unicast in wireless network has been studied extensively in literatures and deployed in practice for years, several important issues about multicast over wireless networks haven't been explored fully. In practice, multicast is a more efficient way to support group communication than unicast or broadcast, as it can transmit packets to destination using fewer network resource, which is critical in wireless networks. Typical wireless multicast applications include group-oriented mobile commerce, military command and control, distance education, and intelligent transportation systems. For a multicast routing, usually a tree with the minimum cost that span the sources and receivers is used because it requires the least network resources than other structures. Finding such minimum cost tree is known to be NP-hard. Thus, some multicast trees with good practical performances have been proposed in the literatures. Unlike unicast problem, as we will show later, if we simply apply VCG mechanism to those commonly used multicast tree structures, we can not guarantee that all wireless devices will follow our prescribed protocols. In this chapter, we discuss how to design truthful non-VCG mechanisms for those multicast structures in *selfish* wireless networks.

The rest of the chapter is organized as follows. In Section 1.2, we review some definitions and priori arts on truthful mechanism design for multicast. In Section 1.3, we present the first strategyproof mechanism for Steiner tree problem (or called multicast). The output of our mechanism (a tree) has a cost within a constant factor of the optimum, and the payment is minimum among any truthful mechanism having this output. In Section 1.4, we show our experimental study of our proposed mechanisms. We conclude the chapter in Section 1.5 by pointing out some possible future works.

1.2 Preliminaries and Priori Art

1.2.1 Wireless Ad hoc Networks

Wireless *ad hoc* networks is emerging as a flexible and powerful wireless architecture that does not rely on a fixed networking infrastructure. Wireless ad hoc networks have received significant attentions over past few years due to their potential applications in various situations such as battlefield, emergency relief and environmental monitor, etc. In a wireless ad hoc network, each mobile node has a transmission range and energy cost. A node v can receive the signal from another node u iff node v is within node u's transmission range. We assume that when node u received a message and then forwarded the message to another node, it would consume node u some energy, which will be categorized as the cost of node u forwarding the data for other nodes. If the receiving node is not within the sender's transmission range, then it must choose some intermediate nodes to repay the message. So unlike the wired networks, all nodes in the wireless ad hoc network should be able to act as a router. On the other aspect, the wireless node usually uses omni-directional antenna, which means that it can use a broadcasting-like manner to distribute the message to all nodes within its transmission range.

Usually, there are two different categories of wireless ad hoc nodes: *fixed* transmission range and *adjustable* transmission range. For *fixed* transmission range nodes, their transmission range have been fixed and can't be adjusted afterward. So there is a directed arc from u to v if node v is in the transmission range of node u. Here the transmission cost depends on node u regardless of the distance between two nodes. Thus the wireless ad hoc network can be considered as a *node weighted graph*, where the weight of each node is its cost to forward a unit data. If all nodes' transmission range is the same, by properly scaling, we can assume all nodes have transmission range 1. Thus, wireless topology can be modelled by a *Unit Disk Graph(UDG)*.

The second type of wireless network is that each wireless node can adjust its transmission range: they can adjust their transmission power to the amount needed to reach the next relay node. The power needed to send a packet from node u to v consists of three part. First, the source node u needs to consume some power to prepare the packet. Second, node u needs consume some power to send the message to v. The power required to support the transmission between u and v not only depends on u but also depends on the geometry distance of u and v. In the literature, it is often assumed that the power needed to support a link uv is $d_u \cdot |uv|^\beta$, where $\beta \in [2, 5]$ depends on the transmission environment, $|uv|$ is the Euclidean distance between u and v, and d_u is a positive number depending on node u only. Finally, when v receives the packet, it needs consume some power to receive, store and then process that packet. Thus, the weight of an edge uv is the power consumed for transmitting packet from u to v plus some possible energy consumed by u and v to process the signal. The wireless network under this model can be considered as a *link weighted graph*: all wireless devices are the vertices of the graph, the weight of each link uv is the total energy cost of communication using link uv.

1.2.2 Algorithm Mechanism Design

In designing efficient, centralized (with input from individual agents) or distributed algorithms and network protocols, the computational agents are typically assumed to be either *correct/obedient* or *faulty* (also called adversarial). Here agents are said to be *correct/obedient* if they follow the protocol correctly; agents are said to be *faulty* if (1) they stop working, or (2) they drop messages, or (3) they act arbitrarily, which is also called *Byzantine failure*, i.e., they may deviate from the protocol in arbitrary ways that harm other users, even if the deviant behavior does not bring them any obvious tangible benefits.

In contrast, as we mentioned before, economists design market mechanisms in which it is assumed that agents are *rational*. The rational agents respond to well-defined incentives and will deviate from the protocol only if it improves its gain. A rational agent is neither correct/obedient nor adversarial.

A standard economic model for analyzing scenarios in which the agents act according to their own self-interests is as follows. There are n agents. Each agent i, for $i \in \{1, \cdots, n\}$, has some private information t^i, called its *type*. The type t^i could be its cost to forward a packet in a network environment, could be its willing payment for a good in an auction environment. Then the set of n agents define a type vector $t = (t^1, t^2, \cdots, t^n)$, which is called the *profile*. There is an output specification that maps each type vector t to a set of allowed outputs. Agent i's preferences over the possible outputs are given by a valuation function v^i that assigns a real number $v^i(t^i, o)$ to each possible output o. Here, notice that the valuation of an agent does not depend on other agents' types. Everything in the scenario is public knowledge except the type t^i, which is a private information to agent i.

Definition 1 ***A Mechanism*** *$M = (A, \mathcal{O}, p)$ defines three functions: a set of strategies A for all agents, an output function $\mathcal{O}$, and a payment function $p = (p^1, \cdots, p^n)$:*

1. *For each agent i, it has a set of strategies A^i. Agent i can only choose a strategy $a \in A^i$.*
2. *For each strategy vector $a = (a^1, \cdots, a^n)$, i.e., the agent i plays a strategy $a^i \in A^i$, the mechanism computes an* output *$o = \mathcal{O}(a^1, \cdots, a^n)$ and a* payment *$p^i = p^i(a)$. Here the payment p^i is the money given to each participating agent i. If $p^i < 0$, it means that the agent has to pay $-p^i$ to participate in the action.*

For an agent i, given the output o and the payment p^i, its utility is $u^i(o, t^i) = v^i(t^i, o) + p^i$. A strategy a^i by an agent i is *dominant* the agent i maximizes its utility *regardless* of whatever other agents do. Considering all different strategies, there will be too many candidate mechanisms, but with the **Revelation Principle**, we only need to focus our attention on these *direct revelation mechanisms*. A mechanism is *direct revelation mechanism* if the types are the strategy space A^i. In this chapter, we will only consider the direct revelation mechanisms. In practice, a mechanism should satisfy the following properties:

1. **Incentive Compatibility (IC)**: The payment function should satisfy the *incentive compatibility*, i.e., for each agent i,

$$v^i(t^i, o(a^{-i}, t^i)) + p^i(a^{-i}, t^i) \geq v^i(t^i, o(a^{-i}, a^i)) + p^i(a^{-i}, a^i).$$

In other words, revealing the type t^i is the *dominating strategy*. If the payment were computed by a strategyproof mechanism, he would have no incentive to lie its type because his overall utility would be no greater than it would have been if he had told the truth.

2. **Individual Rationality (IR)**: It is also called *Voluntary Participation*. For each agent i and any a^{-i} it should have non-negative utilities. In other words, if agent i reveals its true type t^i, then its utility should be non-negative.

3. **Polynomial Time Computability (PC)**: All computation is done in polynomial time. Notice that after every agent declares its type, the mechanism has to compute an output o and a payment vector to all agents. For example, for the family of VCG mechanisms, the output that maximizes the summation of the valuations of all nodes has to be found. When the optimal output cannot be found exactly, the individual agent may have incentives to misreport its type initially.

A mechanism is *strategy-proof* or *truthful* if it satisfies both IR and IC properties. In the remaining of this chapter, we will focus our attention on these *truthful* mechanisms only.

Arguably the most important positive result in mechanism design is what is usually called the generalized Vickrey-Clarke-Groves (VCG) mechanism by Vickrey [18], Clarke [5], and Groves [9]. The VCG mechanism applies to mechanism design maximization problems where the objective function is simply the sum of all agents' valuations and the set of possible outputs is assumed to be finite.

A maximization mechanism design problem is called *utilitarian* if its objective function satisfies that $g(o,t) = \sum_i v^i(t^i, o)$. A direct revelation mechanism $m = (o(t), p(t))$ belongs to the VCG family if (1) the output $o(t)$ computed based on the type vector t maximizes the objective function $g(o,t) = \sum_i v^i(t^i, o)$, and (2) the payment to agent i is

$$p^i(t) = \sum_{j \neq i} v^j(t^j, o(t)) + h^i(t^{-i}).$$

Here $h^i()$ is an arbitrary function of t^{-i} and different agent could have different function $h^i()$ as long as it is defined on t^{-i}. It is proved by Groves [9] that a VCG mechanism is truthful. Green and Laffont [8] proved that, under mild assumptions, VCG mechanisms are the only truthful implementations for utilitarian problems.

An output function of a VCG mechanism is required to maximize the objective function. This makes the mechanism computationally intractable in many cases. Notice that replacing the optimal algorithm with non-optimal approximation usually leads to untruthful mechanisms. In this chapter, we will study how to perform truthful routing for multicast, which is known to be NP-hard and thus VCG mechanisms cannot be applied.

1.2.3 Priori Arts

There are generally two ways to implement the truthful computing: *credit based method* and *incentive based method*.

The first category uses various non-monetary approaches including auditing, system-wide optimal point analysis and some hardwares. Credit based methods have been studied for several years, and most of them are based on the simulation and are heuristic.

In [13], nodes, which agree to relay traffic but do not, are termed as misbehaving. They used *Watchdog* and *Pathrater* to identify misbehaving users and avoid routing through these nodes. The *Watchdog* runs on every node keeping track of how the other nodes behave; the *Pathrater* uses this information to calculate the route with the highest reliability. Notice that this method ignores the reason why a node refused to relay the transit traffics for other nodes. A node will be wrongfully labelled as misbehaving when its battery power cannot support many relay requests and thus refused to relay. It also does not provide any incentives to encourage nodes to relay the message for other nodes.

In [3], Buttyan *et al.* focused on the problem how to stimulate selfish nodes to forward the packets for other nodes. Their approach is based on a so called *nuglet counter* in each node. A node's counter is decreased when sending its own packet, and is increased when forwarding other nodes' packet. All counters should always remain positive. In order to protect the proposed mechanism against misuse, they presented a scheme based on a trusted and tamper resistant hardware module in each node, which generates cryptographically protected security headers for packets and maintains the nuglet counters of the nodes. They also studied the behavior of the proposed mechanism analytically and by means of simulations, and showed that it indeed stimulates the nodes for packet forwarding.

In [4], they still use a nugget counter to store the nuglets and besides that they use a fine which decreases the nugget counter to prevent the node from not relaying the packet. They use the *packet purse model* to discourage the user to send useless traffic and overload the network. The basic idea presented in [4] is similar to [3] but different in the implementation.

In [16], two acceptance algorithms are proposed. These algorithms are used by the network nodes to decide whether to relay traffic on a per session basis. The goal of them is to balance [1] the energy consumed by a node in relaying traffics for others with energy consumed by other nodes to relay its traffic and to find an optimal trade-off between energy consumption and session blocking probability. By taking decisions on a per session basis, the per packet processing overhead of previous schemes is eliminated. In [17], a distributed and scalable acceptance algorithm called GTFT is proposed. They proved that GTFT results in Nash equilibrium and the system converges to the rational and optimal operating point. Notice that, they assumed that each path is h hops long and the h relay nodes are chosen with equal probability from the remaining $n-1$ nodes, which may be unrealistic.

In [15], Salem *et al.* presented a charging and rewarding scheme for packet forwarding in multi-hop cellular networks. In their network model, there is a base-station to forward the packets. They use symmetric cryptography to cope with the lying. To count several possible attacks, it pre-charges some nodes and then refunds them only if a proper acknowledgment is received. Their basic payment scheme is still based on nuglets.

In [11] Jakobsson *et al.* described an architecture for fostering collaboration between selfish nodes of multi-hop cellular networks. Based on this architecture, they provided mechanisms based on per packet charge to encourage honest behavior and to discourage dishonest

[1] It is impossible to strictly balance the number of packets a node has relayed for other nodes and the number of packets of this node relayed by other nodes since, in a wireless ad hoc network, majority of the packet transmissions are relayed packets. For example, consider a path of h hops. $h-1$ nodes on the path relay the packets for others. If the average path length of all routes is h, then $1-1/h$ fraction of the transmissions are transit traffics.

behavior. In their approach, all packet originators attach a payment token to each packet, and all intermediaries on the packet's path to the base station verify whether this token corresponds to a special token called *winning ticket*. Winning tickets are reported to nearby base stations at regular intervals. The base stations, therefore, receive both reward claims (which are forwarded to some accounting center), and packets with payment tokens. After verifying the validity of the payment tokens, base stations send the packets to their desired destinations, over the backbone network. The base stations also send the payment tokens to an accounting center. Their method also involves some traditional security method including auditing, node abuse detection and encryption etc.

The *Incentive based methods* borrow some ideas from the micro-economic and game-theoretic world, which involve the monetary transfer. The key result of this category is that all nodes won't deviate from their normal activities because they will benefit most when they reveal their true cost, even knowing all other nodes' true costs. We can thus achieve the optimal system performance. This idea has been introduced by Nisan and Ronen in [14] and is known as the algorithm mechanism design.

In [14], Nisan and Ronen provided a polynomial-time strategyproof mechanism for optimal unicast route selection in a centralized computational model. In their formulation, the network is modelled as an abstract graph $G = (V, E)$. Each edge e of the graph is an agent and has a private type t_e, which represents the cost of sending a message along this edge. The mechanism-design goal is to find a Least Cost Path (LCP) $\mathsf{LCP}(x, y)$ between two designated nodes x and y. The valuation of an agent e is $-t_e$ if the edge e is part of the path $\mathsf{LCP}(x, y)$ and 0 otherwise. Nisan and Ronen used the VCG mechanism for payment. The payment to an agent e is $D_{G-\{e\}}(x, y) - D_G(x, y)$, where $D_{G-\{e\}}(x, y)$ is the cost of the LCP through G when edge e is not presented and $D_G(x, y)$ is the cost of the least cost path $\mathsf{LCP}(x, y)$ through G. Clearly, there must have two link disjoint paths connecting x and y to prevent the monopoly. The result in [14] can be easily extended to deal with wireless unicast problem for arbitrary pair of terminals.

Feigenbaum *et. al* [6] then addressed the truthful low cost routing in a different network model. They assume that each node k incurs a transit cost c_k for each transit packet it carries. For any two nodes i and j of the network, $T_{i,j}$ is the intensity of the traffic (number of packets) originating from i and destined for node j. Their strategyproof mechanism again is essentially the VCG mechanism. They gave a distributed method such that each node i can compute a payment $p_{ij}^k > 0$ to node k for carrying the transit traffic from node i to node j if node k is on the LCP $\mathsf{LCP}(i, j)$. Anderegg and Eidenbenz [1] recently proposed a similar routing protocol for wireless ad hoc networks based on VCG mechanism again. They assumed that each link has a cost and each node is a selfish agent.

For multicast flow, Feigenbaum *et. al* [7] assumed that there is *fixed* multicast infrastructure, given any set of receivers $Q \subset V$, connects the source node to the receivers. Additionally, for each user $q_i \in Q$, they assumed a *fixed* path from the source to it, determined by the multicast routing infrastructure. Then for every subset R of receivers, the delivery tree $T(R)$ is merely the union of the fixed paths from the source to the receivers R. They also assumed that there is a link cost associated with each communication link in the network and the link cost is *known* to everyone. For each receiver q_i, there is a valuation w_i that this user values the reception of the data from the source. This information w_i is only known to q_i. User q_i will report a number w_i', which is the amount of money he/she is willing to pay to receive the data. The source node then selects a subset $R \subset Q$ of receivers to maximize the difference $\sum_{i \in R} w_i' - C(R)$, where $C(R)$ is the cost of the multicast tree $T(R)$ to send data

to all nodes in R. The approach of fixing the multicast tree is relatively simple to implement but could not model the greedy nature of all network terminals in the network.

1.2.4 Problem Statement and Network Model

In this chapter, we consider a wireless ad hoc network composed of n selfish nodes $V = \{v_1, v_2, \cdots, v_n\}$. Every node v_i has a fixed transmission range r_i, node u and v can communicate with each other if and only if $|v_i v_j| \leq \min\{r_i, r_j\}$. When node v_i send a packet to one of its neighbors, say v_j, all v_i's neighbors can receive this packet. Thus, every node will broadcast its packet. We assume each node v_i has a private cost c_i to broadcast a unit data(the unit data could be 1 Byte or 1 MegaByte). In this chapter, we model this wireless ad hoc network as a node weighted graph $G = (V, E, c)$ where V is the set if wireless nodes, and $e = v_i v_j \in E$ if and only if v_i and v_j can communicate with each other. Here $c = \{c_1, c_2, \cdots, c_n\}$ is the cost profile of all nodes. Notice here the graph is undirected.

Based on the node weighted graph, we now define the multicast problem as follows. Given a set of receivers $Q = \{q_0, q_1, q_2, \cdots, q_{r-1}\} \subset V$, when selects node $q_i \in Q$ as the source, the multicast problem is to find a tree $T \subset G$ spanning all receiving terminals Q. For simplicity, we assume that q_0 is the source of the multicast. Each node v_i is required to declare a cost d_i of relaying the message. Based on the declared cost profile $d = \{d_1, d_2, \cdots, d_n\}$, the source node constructs the multicast tree and decide the payment for each node. It is well-known [10, 12] that it is NP-hard to find the minimum cost multicast tree when given an arbitrary node weighted graph G, and it is at least as hard to approximate as the set cover problem. Klein and Ravi [12] showed that it can be approximated within $O(\ln r)$, where r is the number of receivers. The utility of an agent is its payment received, minus its cost if it is selected in the multicast tree. Instead of reinventing the wheels, we will still use the previously proposed structures for multicast as the output of our mechanism. Given a multicast tree, we will study the designing of strategyproof payment schemes based on these trees.

Given a graph G, we use $\omega(G)$ to denote the total cost of all nodes in this network. If we change the cost of any agent i (link e_i or node v_i) to c_i', we denote the new network as $G' = (V, E, c|^i c_i')$, or simply $c|^i c_i'$. If we remove one agent i from the network, we denote it as $c|^i \infty$. Denote $G \backslash e_i$ as the network without link e_i, and denote $G \backslash v_i$ as the network without node v_i and all its incident links. For the simplicity of notation, we will use the cost vector c to denote the network $G = (V, E, c)$ if no confusion is caused.

1.3 Strategyproof Multicast

In this section, we discuss in detail how to conduct truthful multicast when the network is modelled by a node weighed communication graph. We specifically study the following three structures: least cost path star(LCPS), virtual minimum spanning tree (VMST) and node weighted Steiner tree (NST). In practice, for various applications, usually receivers/senders in the same multicast group usually belong to the same organization or company, so their behavior can be expected to be cooperative instead of uncooperative. Thus, we assume that every receiver will relay the packet for other receivers for free.

1.3.1 Strategyproof Mechanism Based on LCPS

1.3.2 Least Cost Path Star

Given a network modelled by the graph G, a source node s, and a set of r receivers Q, the least cost path star is the union of all r shortest paths from the receiver to each of the receivers in Q. In practice this is one of the most widely used methods of constructing the multicast tree since it takes advantage of the unicast routing information collected by Distance-Vector Algorithm or Link-State Algorithm. Notice that, although here we only discuss the using of least cost path star for the node weighted case, all results we presented in this subsection can be extended to the link weighted scenario without any difficulty, when each link will incur a cost when transmitting data.

Constructing LCPS

For each receiver $q_i \neq s$, we compute the shortest path (least cost path), denoted by $\mathsf{LCP}(s, q_i, d)$, from the source s to q_i under the reported cost profile d. The union of all least cost paths from the source to receivers is called *least cost path star*, denoted by $LCPS(d)$. Clearly, we can construct LCPS in time $O(n \log n + m)$. The remaining part is how to design a truthful payment scheme while using LCPS as output.

VCG mechanism on LCPS is not strategyproof

Intuitively, we would like to use the VCG payment scheme in conjunction with the LCPS tree structure as follows. The payment $p_k(d)$ to every node v_k is

$$p_k(d) = \omega(LCPS(d|^k\infty)) - \omega(LCPS(d)) + d_k.$$

We show by an example that the above payment scheme is not strategyproof. Figure **??** illustrates such an example where node v_2 will have a negative utility when it reveals its true cost.

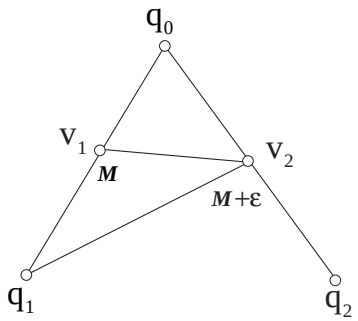


Figure 1.1: The cost of terminals are $v_1 = M$ and $v_2 = M + \epsilon$.

Notice $\omega(LCPS(c)) = 2M + \epsilon$ and $\omega(LCPS(c|^1\infty)) = M + \epsilon$. If v_1 reveals its true cost, its payment is $p_1(c) = \omega(LCPS(c|^1\infty)) - \omega(LCPS(c)) + M = M + \epsilon - (2M + \epsilon) + M = 0$. Thus, it utility is $p^1(c) - C_1 = 0 - M < 0$, which violates the IR property.

Strategyproof mechanism on LCPS

Now, we describe our strategyproof mechanism that does not rely on VCG payment. For each receiver $q_i \neq s$, we compute the least cost path from the source s to q_i, and compute a payment $p_k^i(d)$ to every node v_k on the $\mathsf{LCP}(s, q_i, d)$ using the scheme for unicast

$$p_k^i(c) = d_k + |\mathsf{LCP}(s, q_i, d|^k\infty)| - |\mathsf{LCP}(s, q_i, d)|.$$

Here $|\mathsf{LCP}(s, q_i, d)|$ denotes the total cost of the least cost path $\mathsf{LCP}(s, q_i, d)$. The total payment to a link v_k is then

$$p_k(d) = \max_{q_i \in Q} p_k^i(d) \tag{1.1}$$

Theorem 1 *Payment (1.1) based on LCPS is truthful and it is minimum among all truthful payments based on LCPS.*

PROOF. Clearly, when node v_k reports its cost truthfully, it has non-negative utility, i.e., the payment scheme satisfies the IR property. In addition, since the payment scheme for unicast is truthful, so v_k cannot lie its cost to increase its payment $p_k^i(c)$ based on $\mathsf{LCP}(s, q_i, d)$. Thus, it cannot increase $\max_{q_i \in Q} p_k^i(c)$ by lying its cost. In other words, our payment scheme is truthful.

We then show that the above payment scheme pays the minimum among all strategyproof mechanisms using LCPS as the output. Before showing the optimality of our payment scheme, we give some definitions first. Consider all paths from source node s to a receiver q_i, they can be divided into two categories: with node v_k or without node v_k. The path with the minimum length among these paths with node v_k is denoted as $\mathsf{LCP}_{v_k}(s, q_i, d)$; and the path with the minimum length among these paths without edge v_k is denoted as $\mathsf{LCP}_{-v_k}(s, q_i, d)$.

Assume that there is another payment scheme $\tilde{p}$ that pays less for a link v_k in a network G under a cost profile d. Let $\delta = p_k(d) - \tilde{p}_k(d)$, then $\delta > 0$. Without loss of generality, assume that $p_k(d) = p_k^i(d)$. Thus, node v_k is on $\mathsf{LCP}(s, q_i, d)$ and the definition of $p_k^i(d)$ implies that

$$|\mathsf{LCP}_{-v_k}(s, q_i, d)| - |\mathsf{LCP}(s, q_i, d)| = p_k(d) - d_k.$$

Then consider another cost profile $d' = d|^k(p_k(d) - \frac{\delta}{2})$ where the true cost of node v_k is $p_k(d) - \frac{\delta}{2}$. Under profile d', since $|\mathsf{LCP}_{-v_k}(s, q_i, d')| = |\mathsf{LCP}_{-e_k}(s, q_i, d)|$, we have

$$\begin{aligned}
|\mathsf{LCP}_{v_k}(s, q_i, d')| &= |\mathsf{LCP}_{v_k}(s, q_i, d|^k 0)| + p_k(d) - \frac{\delta}{2} \\
&= |\mathsf{LCP}_{v_k}(s, q_i, d)| + p_k(d) - \frac{\delta}{2} - d_k \\
&= |\mathsf{LCP}(s, q_i, d)| + p_k(d) - \frac{\delta}{2} - d_k \\
&= |\mathsf{LCP}_{-v_k}(s, q_i, d)| - \frac{\delta}{2} \\
&< |\mathsf{LCP}_{-v_k}(s, q_i, d)| = |\mathsf{LCP}_{-v_k}(s, q_i, d')|
\end{aligned}$$

Thus, $v_k \in LCPS(d')$. From the following Lemma 1, we know that the payment to node v_k is the same for cost profile d and d'. Thus, the utility of link v_k under the profile d' by the payment scheme $\tilde{p}$ becomes $\tilde{p}_k(d') - c_k = \tilde{p}_k(d) - c_k = \tilde{p}_k(d) - (p_k(d) - \frac{\delta}{2}) = -\frac{\delta}{2} < 0$. In other words, under the profile d', when the node e_k reports its true cost, it gets a negative utility under payment scheme $\tilde{p}$. Thus, $\tilde{p}$ is not strategyproof. This finishes our proof. □

Lemma 1 *If a mechanism with output T and the payment function $\tilde{p}$ is truthful, then for every node v_k in network, if $v_k \in T$ then payment function $\tilde{p}_k(d)$ should be independent of d_k.*

PROOF. We prove it by contradiction. Suppose that there exists a truthful payment scheme such that $\tilde{p}_k(d)$ depends on d_k. There must exist two valid declared costs x_1 and x_2 for node v_k such that $x_1 \neq x_2$ and $\tilde{p}_k(d|^k x_1) \neq \tilde{p}_k(d|^k x_2)$. Without loss of generality we assume that $\tilde{p}_k(d|^k x_1) > \tilde{p}_k(d|^k x_2)$. Now consider the situation when node v_k has an actual cost $c_k = x_2$. Obviously, node v_k can lie its cost as x_2 to increase his utility, which violates the incentive compatibility (IC) property. This finishes the proof. □

Notice that the payment based on $p_k(c) = \min_{q_i \in Q} p_k^i(c)$ is not truthful since a link may lie its cost upward so it can discard some low payment from some receiver. In addition, the payment $p_k(c) = \sum_{q_i \in Q} p_k^i(c)$ is *not* truthful either.

1.3.3 Strategyproof Mechanism Based on VMST

Constructing VMST

We first describe our method to construct the virtual minimum spanning tree.

Algorithm 1 ***Virtual MST Algorithm***

1. First, calculate the pairwise least cost path $\mathsf{LCP}(q_i, q_j, d)$ between any two terminals $q_i, q_j \in Q$ when the cost vector is d.
2. Construct a virtual complete link weighted network $K(d)$ using Q (including the source node here) as its terminals, where the link q_iq_j corresponds to the least cost path $\mathsf{LCP}(q_i, q_j, d)$, and its weight $w(q_iq_j)$ is the cost of the path $\mathsf{LCP}(q_i, q_j, d)$, i.e., $w(q_iq_j) = |\mathsf{LCP}(q_i, q_j, d)|$.
3. Build the minimum spanning tree (MST) on $K(d)$. The resulting MST is denoted as $VMST(d)$.
4. For every virtual link q_iq_j in $VMST(d)$, we find the corresponding least cost path $\mathsf{LCP}(q_i, q_j, d)$ in the original network. Combining all these paths can generate a subgraph of G, say $VMSTO(d)$.
5. All nodes on $VMSTO(d)$ will relay the packets.

Notice that a terminal v_k is on $VMSTO(d)$ iff v_k is on some virtual links in the $VMST(d)$, so we can focus our attention on these terminals in $VMST(d)$. It is not very difficult to show that the cost of VMST could be very large compared to the optimal. But when all nodes have the same transmission ranges in the original wireless ad hoc network, which can be modelled as UDG, following theorem shows that the virtual minimum spanning tree can approximate cost of the optimal tree within a constant factor.

Theorem 2 *$VMST(G)$ is a 5-approximation of the optimal solution in terms the total cost if the wireless ad hoc network is modelled by a unit disk graph.*

PROOF. Assume that the optimal solution is a tree called T_{opt}. Let $V(T_{opt})$ be the set of nodes used in the tree T_{opt}. Clearly, $\omega(T_{opt}) = \sum_{v_i \in V(T_{opt})} c_i$. Similarly, for any spanning tree T of $K(G, Q)$, we define $\omega(T) = \sum_{e \in T} w(e)$. Following we will prove $5 \cdot \omega(T_{opt}) \geq \omega(VMST(G))$.

First, for all nodes in T_{opt}, when disregarding the node weight, there is a spanning tree T'_{opt} on $V(T_{opt})$ with node degree at most 5 since the wireless network is modelled by a unit disk graph. This is due to a well-known fact that there is an Euclidean minimum spanning tree with the maximum node degree at most 5 for any set of two-dimensional points. Note here we only need to know the existence of T'_{opt}, we do not require to construct such spanning tree explicitly. Obviously, $\omega(T_{opt}) = \omega(T'_{opt})$. Thus, tree T'_{opt} is also an optimal solution. with maximal node degree degree at most 5.

For spanning tree T'_{opt}, we root it at an arbitrary node and duplicate every link in T'_{opt} (the resulting structure is called DT'_{opt}). Clearly, every node in DT'_{opt} has even degree now. Thus, we can find an Euler circuit, denoted by $EC(DT'_{opt})$, that visits every vertex of DT'_{opt} and uses every edge of DT'_{opt} exactly once, which is equivalent to say that every edge in $T'_{opt}(G)$ is used exactly twice. Consequently, we know that every node v_k in $V(T_{opt})$ is used

exactly $deg_{T'_{opt}}(v_k)$ times. Here $deg_G(v)$ denotes the degree of a node v in a graph G. Thus, the total weight of the Euler circuit is at most 5 times of the weight $\omega(T'_{opt})$, i.e.,

$$\omega(EC(DT'_{opt})) \leq 5 \cdot \omega(T'_{opt}).$$

Notice that here if a node v_k appears multiple times in $EC(DT'_{opt})$, its weight is also counted multiple times in $\omega(EC(DT'_{opt}))$.

If we walk along $EC(DT'_{opt})$, we visit all receivers, and length of any subpath between receivers q_i and q_j is no smaller than $|\mathsf{LCP}(q_i, q_j, G)|$. Thus, the cost of $EC(DT'_{opt})$ is at least $\omega(VMST(G))$ since $VMST(G)$ is the minimum spanning tree spanning all receivers and the cost of the edge q_iq_j in $VMST(G)$ corresponds the path with the least cost $|\mathsf{LCP}(q_i, q_j, G)|$. In other words,

$$\omega(EC(DT'_{opt})) \geq \omega(VMST(G)).$$

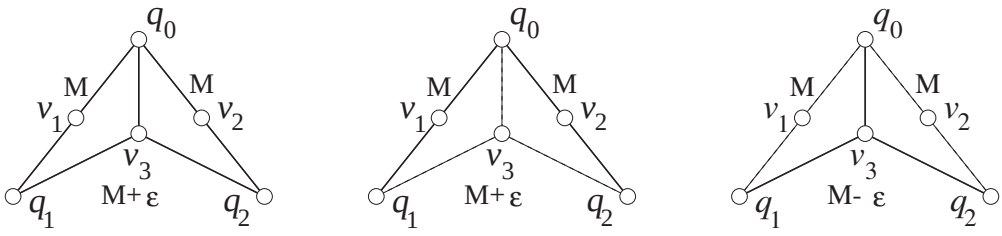
Consequently, we have

$$\omega(VMST(G)) \leq \omega(EC(DT'_{opt})) \leq 5 \cdot \omega(T'_{opt}).$$

This finishes the proof. □

VCG mechanism on VMST is not strategy-proof

In this subsection, we show that a simple application of VCG mechanism on VMST is not strategy-proof. Figure 1.2 illustrates such an example where terminal v_3 can lie its cost to improve its utility when output is VMST. The payment to terminal v_3 is 0 and its utility is also 0 if it reports its cost truthfully. The total payment to terminal v_3 when v_3 reported a cost $d_3 = M - \epsilon$ is $\omega(VMST(c|^3\infty)) - \omega(VMST(c|^3d_3)) + d_3 = 2M - (M - \epsilon) + M - \epsilon = 2M$ and the utility of terminal v_3 becomes $u_3(c|^3d_3) = 2M - (M + \epsilon) = M - \epsilon$, which is larger than $u_3(c) = 0$. Thus, VCG mechanism based on VMST is not strategy-proof.



(a) Graph G (b) VMST (c) VMST with lie

Figure 1.2: The cost of terminals are $c_4 = c_5 = M$ and $c_3 = M + \epsilon$.

Strategyproof Mechanism on VMST

Before discussing the strategyproof mechanism based on VMST, we give some related definitions first. Given a spanning tree T and a pair of terminals p and q on T, clearly there is a

unique path connecting them on T. We denote such path as $\Pi_T(p,q)$, and the edge with the maximum length on this path as $LE(p,q,T)$. For simplicity, we use $LE(p,q,d)$ to denote $LE(p,q,VMST(d))$ and use $LE(p,q,d|^k d'_k)$ to denote $LE(p,q,VMST(d|^k d'_k))$.

Following is our truthful payment scheme when the output is the multicast tree $VMST(d)$.

Algorithm 2 *Truthful payment scheme based on VMST*

1. For every terminal $v_k \in V \setminus Q$ in G, first calculate $VMST(d)$ and $VMST(d|^k\infty)$ according to the terminals' declared costs vector d.
2. For any edge $e = q_iq_j \in VMST(d)$ and any terminal $v_k \in \mathsf{LCP}(q_i,q_j,d)$, we define the payment to terminal v_k based on the virtual link q_iq_j as follows:

$$p_k^{ij}(d) = |LE(q_i,q_j,d|^k\infty)| - |\mathsf{LCP}(q_i,q_j,d)| + d_k.$$

Otherwise, $p_{ij}^k(d)$ is 0. The final payment to terminal v_k based on $VMST(d)$ is

$$p_k(d) = \max_{q_iq_j \in VMST(d)} p_k^{ij}(d). \tag{1.2}$$

Theorem 3 *Our payment scheme (1.2) is strategyproof and minimum among all truthful payment schemes based on VMST structure.*

Instead of proving Theorem 3, we prove Theorem 4, Theorem 5 and Theorem 6 in the remaining of this subsection.

Before the proof of Theorem 3, we give some related notations and observations. Considering the graph $K(d)$ and a node partition $\{Q_i, Q_j\}$ of Q, if an edge's two end-nodes belong to different node set of the partition, we call it a *bridge*. All bridges q_sq_t over node partition Q_i, Q_j in the graph $K(d)$ satisfying $v_k \notin \mathsf{LCP}(q_s,q_t,d)$ form a bridge set $B^{-v_k}(Q_i,Q_j,d)$. Among them, the bridge with the minimum length is denoted as $MB^{-v_k}(Q_i,Q_j,d)$ when the nodes' declared cost vector is d. Similarly, All bridges q_sq_t over node partition Q_i, Q_j in the graph $K(d)$ satisfying $v_k \in \mathsf{LCP}(q_s,q_t,d)$ form a bridge set $B^{v_k}(Q_i,Q_j,d)$. The bridge in $B^{v_k}(Q_i,Q_j,d)$ with the minimum length is denoted as $BM^{v_k}(Q_i,Q_j,d)$. Obviously, we have

$$BM(Q_i,Q_j,d) = \min\{BM^{v_k}(Q_i,Q_j,d), BM^{-v_k}(Q_i,Q_j,d)\}.$$

We then state our main theorems for the payment scheme discussed above.

Theorem 4 *Our payment scheme satisfies IR.*

PROOF. First of all, if terminal v_k is not chosen as relay terminal, then its payment $p_k(d|^k c_k)$ is clearly 0 and its valuation is also 0. Thus, its utility $u_k(d|^k c_k)$ is 0.

When terminal v_k is chosen as a relay terminal when it reveals its true cost c_k, we have $|LE(q_i, q_j, d|^k\infty)| \geq |\mathsf{LCP}(q_i, q_j, d|^k c_k)|$. This is due to the following observation: For any cycle C in a graph G, assume e_c is the longest edge in the cycle, then $e_c \notin MST(G)$. The lemma immediately follows from

$$p_{ij}^k(d|^k c_k) = |LE(q_i, q_j, d|^k\infty)| - |\mathsf{LCP}(q_i, q_j, d|^k c_k)| + c_k > c_k.$$

This finishes the proof. □

From the definition of the incentive compatibility (IC), we assume that the d_{-k} is fixed throughout the proof. For our convenience, we will use $G(d_k)$ to represent the graph $G(d|^k d_k)$. We first prove a series of lemmas that will be used to prove that our payment scheme satisfies IC.

Lemma 2 *If $v_k \in q_iq_j \in VMST(d)$, then $p_k^{ij}(d)$ doesn't depend on d_k.*

Proof. Remember that the payment based on a link q_iq_j is $p_k^{ij}(d) = |LE(q_i, q_j, d|^k\infty)| - |\mathsf{LCP}(q_i, q_j, d)| + d_k$. The first part $LE(q_i, q_j, d|^k\infty)$ is the longest edge of the unique path from q_i to q_j on tree $VMST(d|^k\infty)$. Clearly, it is independent of d_k. Now considering the second part $\mathsf{LCP}(q_i, q_jd) - d_k$. From the assumption we know that $v_k \in \mathsf{LCP}(q_i, q_j, d)$, so the path $\mathsf{LCP}(q_i, q_j, d)$ remains the same regardless of v_k's declared cost d_k. Thus, the summation of all terminals' cost on $\mathsf{LCP}(q_i, q_j, d)$ except terminal v_k equals to

$$|\mathsf{LCP}(q_i, q_j, d|^k 0)| = |\mathsf{LCP}(q_i, q_j, d)| - d_k.$$

In other words, the second part is also independent of d_k. Now we can write the payment to a terminal v_k based on an edge q_iq_j as following:

$$p_k^{ij}(d) = |LE(q_i, q_j, d|^k\infty)| - |\mathsf{LCP}(q_i, q_j, d|^k 0)|,$$

Here terminal $v_k \in \mathsf{LCP}(q_i, q_j, d)$ and $q_iq_j \in VMST(d)$. □

If a terminal v_k lies its cost c_k upward, we denote the lied cost as $\overline{c_k}$. Similarly, if terminal v_k lies its cost c_k downward, we denote the lied cost as $\underline{c_k}$. Let $E_k(d_k)$ be the set of edges q_iq_j such that $v_k \in \mathsf{LCP}(q_i, q_j, d)$ and $q_iq_j \in VMST(d)$ when terminal v_k declares a cost d_k. From Lemma 2, the non-zero payment to v_k is defined based on $E_k(d_k)$. Following lemma reveals the relationship between d_k and $E_k(d_k)$. The proof of the lemma is omitted due to the simplicity.

Lemma 3 *$E_k(d_k) \subseteq E_k(\underline{d'_k})$ when $d'_k \leq d_k$.*

We now state the proof that the payment scheme 1.2 satisfies IC.

Theorem 5 *Our payment scheme satisfies the incentive compatibility (IC).*

PROOF. For terminal v_k, if it lies its cost from c_k to $\overline{c_k}$, then $E_k(\overline{c_k}) \subseteq E_k(c_k)$, which implies that payment

$$\begin{aligned} p_k(d|^k\overline{c_k}) &= \max_{q_iq_j \in E_k(\overline{c_k})} p_k^{ij}(d|^k\overline{c_k}) \\ &\leq \max_{q_iq_j \in E_k(c_k)} p_k^{ij}(d|^k c_k) = p^k(d|^k c_k). \end{aligned}$$

Thus, terminal v_k won't lies it cost upward, so we focus our attention on the case when terminal v_k lies its cost downward.

From Lemma 3, we know that $E_k(c_k) \subseteq E_k(\underline{c_k})$. Thus, we only need to consider the payment based on edges in $E_k(\underline{c_k}) - E_k(c_k)$. For edge $e = q_iq_j \in E_k(\underline{c_k}) - E_k(c_k)$, let $q_I^kq_J^k = LE(q_i, q_j, d|^k\infty)$ in the spanning tree $VMST(d|^k\infty)$. If we remove the edge $q_I^kq_J^k$, we have a vertex partition $\{Q_I^k, Q_J^k\}$, where $q_i \in Q_I^k$ and $q_j \in Q_J^k$. In the graph $K(d)$, we consider the bridge $BM(Q_I^k, Q_J^k, d)$ whose weight is minimum when the terminals cost vector is d. There are two cases need to be considered about $BM(Q_I^k, Q_J^k, d)$: 1) $v_k \notin BM(Q_I^k, Q_J^k, d|^k c_k)$ or 2) $v_k \in BM(Q_I^k, Q_J^k, d|^k c_k)$. We discuss them individually.

Case 1: $v_k \notin BM(Q_I^k, Q_J^k, d|^k c_k)$. In this case, edge $q_I^kq_J^k$ is the minimum bridge over Q_I^k and Q_J^k. In other words, we have $|LE(q_i, q_j, |^k\infty)| \leq |\mathsf{LCP}(q_i, q_j, d|^k c_k)|$. Consequently

$$\begin{aligned} p_k^{ij}(d|^k\underline{c_k}) &= |LE(q_i, q_j, d|^k\infty)| - |\mathsf{LCP}(q_i, q_j, d|^k\underline{c_k})| + \underline{c_k} \\ &= |LE(q_i, q_j, d|^k\infty)| - |\mathsf{LCP}(q_i, q_j, d|^k c_k)| + c_k \\ &\leq c_k, \end{aligned}$$

which implies that v_k will not get benefit from lying its cost downward.

Case 2: $v_k \in BM(Q_I^k, Q_J^k, d|^k c_k)$. From the assumption that $q_iq_j \notin VMST(G(d|^k c_k))$, we know edge q_iq_j cannot be $BM(Q_I^k, Q_J^k, d|^k c_k)$. Thus, there exists an edge $q_sq_t \neq q_iq_j$ such that $v_k \in \mathsf{LCP}(q_s, q_t, d|^k c_k)$ and $q_sq_t = BM(Q_I^k, Q_J^k, d|^k c_k)$. This guarantees that $q_sq_t \in VMST(d|^k c_k)$.

Obviously, q_sq_t cannot appear in the same set of Q_I^k or Q_J^k. Thus, $q_I^kq_J^k$ is on the path from q_s to q_t in graph $VMST(d|^k\infty)$, which implies that $|\mathsf{LCP}(q_I^k, q_J^k, d|^k\infty)| = |LE(q_i, q_j, d|^k\infty)| \leq |LE(q_s, q_t, d|^k\infty)|$. Using Lemma 3, we have $\mathsf{LCP}(q_s, q_t, d|^k\underline{c_k}) \in VMST(d|^k\underline{c_k}))$. Thus,

$$\begin{aligned} p_k^{ij}(d|^k\underline{c_k}) &= |LE(q_i, q_j, d|^k\infty)| - |\mathsf{LCP}(q_i, q_j, d|^k\underline{c_k})| + \underline{c_k} \\ &= |LE(q_i, q_j, d|^k\infty)| - |\mathsf{LCP}(q_i, q_j, d|^k c_k)| + c_k \\ &\leq |LE(q_s, q_t, d|^k\infty)| - |\mathsf{LCP}(q_i, q_j, d|^k c_k)| + c_k \\ &\leq |LE(q_s, q_t, d|^k\infty)| - |\mathsf{LCP}(q_s, q_t, d|^k c_k)| + c_k \\ &= p_k^{st}(d|^k c_k) \end{aligned}$$

This inequality concludes that even if v_k lies its cost downward to introduce some new edges in $E_k(\underline{c_k})$, the payment based on these newly introduced edges is not larger than the

payment on some edges already contained in $E_k(c_k)$. In summary, node v_i don't have the incentive to lie its cost upward or downward, which proves the IC property. □

Before proving Theorem 6, we prove the following lemma regarding all truthful payment schemes based on VMST.

Lemma 4 *If $v_k \in VMST(d|^k c_k)$, then as long as $d_k < p_k(d|^k c_k)$ and d^{-k} fixed, $v_k \in VMST(d)$.*

PROOF. Again, we prove it by contradiction. Assume that $v_k \notin VMST(d)$. Obviously, $VMST(d) = VMST(d|^k \infty)$. Assume that $p_k(d|^k c_k) = p_k^{ij}(d|^k c_k)$, i.e., its payment is computed based on edge $q_i q_j$ in $VMST(d|^k c_k)$. Let $q_I q_J$ be the $LE(q_i, q_j, d|^k \infty)$ and $\{Q_i, Q_j\}$ be the vertex partition introduced by removing edge $q_I q_J$ from the tree $VMST(d|^k \infty)$, where $q_i \in Q_i$ and $q_j \in Q_j$. The payment to terminal v_k in $VMST(d|^k c_k)$ is $p_k(d|^k c_k) = |\mathsf{LCP}(q_I, q_J, d|^k \infty)| - c_{ij}^{v_k}$, where $c_{ij}^{v_k} = |\mathsf{LCP}(q_i, q_j, d|^k 0|$. When v_k's declare its cost as d_k, the length of the path $\mathsf{LCP}(q_i, q_j, d)$ becomes $c_{ij}^{v_k} + d_k = |\mathsf{LCP}(q_I, q_J, d|^k \infty)| - p_k(d|^k c_k) + d_k < |\mathsf{LCP}(q_I, q_J, d|^k \infty)|$.

Now consider the spanning tree $VMST(d)$. We have assumed that $v_k \notin VMST(d)$, i.e., $VMST(d) = VMST(d|^k \infty)$. Thus, among the bridge edges over Q_i, Q_j, edge $q_I q_J$ has the least cost when graph is $G \backslash v_k$ or $G(d|^k d_k)$. However, this is a contradiction to we just proved: $|\mathsf{LCP}(q_i, q_j, d|^k d_k)| < |\mathsf{LCP}(q_I, q_J, d|^k \infty)|$. This finishes the proof. □

We now are ready to show that our payment scheme is optimal among all truthful mechanisms using VMST.

Theorem 6 *Our payment scheme is the minimum among all truthful payment schemes based on the VMST structure.*

PROOF. We prove it by contradiction. Assume that there is another truthful payment scheme, say $\mathcal{A}$, based on VMST, whose payment is smaller than our payment for a terminal v_k under a cost profile d. Assume that the payment calculated by $\mathcal{A}$ for terminal v_k is $\tilde{p}_k(d) = p_k(d) - \delta$, where $p_k(d)$ is the payment calculated by our algorithm and $\delta > 0$.

Now consider another profile $d|^k d'_k$, where the terminal v_k has the true cost $c_k = d'_k = p^k(d) - \frac{\delta}{2}$. From Lemma 4, we know that v_k is still in $VMST(d|^k d'_k)$. Using Lemma 1, we know that the payment for terminal v_k using algorithm $\mathcal{A}$ is $p_k(c) - \delta$, which is independent of terminal v_k's declared cost. Notice that $d_k = p_k(d) - \frac{\delta}{2} > p_k(d) - \delta$. Thus, terminal v_k has a negative utility under the payment scheme $\mathcal{A}$ when node v_k reveals it true cost under cost profile $d|^k d'_k$, which violates the incentive compatibility (IC). This finishes the proof. □

By summarizing Theorem 4, Theorem 5 and Theorem 6, we get Theorem 3.

1.3.4 Strategyproof Mechanism Based on Spider

For a general node weighted network, in the worst case, the cost of the structure LCPS and VMST could be $\theta(n)$ times of cost of the optimal tree. It is known (e.g., [12],[10]) that it is NP-hard to find the minimum cost multicast tree when given an arbitrary node weighted graph G, and it is at least as hard to approximate as the set cover problem. Klein and Ravi [12] showed that it can be approximated within $O(\ln r)$, where r is the number of receivers, which is within a small constant factor of the best achievable approximation ratio among all polynomial time computable tree if $N \neq NP$.

Constructing the spider

Here, we review the method used in [12] to find a node weighted Steiner tree (NST). In [12], the authors used a special structure called *spider* to approximate the optimal solution. A spider is defined as a tree having at most one node of degree more than two. Such a node (if exists) is called the center of the spider. Each path from the center to a leaf is called a *leg*. The *cost* of a spider S is defined as the sum of the cost of all nodes in spider S, denotes as $\omega(S)$. The number of terminals or *legs* of the spider is denoted by $t(S)$, and ratio of a spider S is defined as

$$\rho(S) = \frac{\omega(S)}{t(S)}.$$

Contraction of a spider S is the operation of contracting all vertices of S to form one virtual terminal and connect this virtual terminal to each vertex v when uv is a link before the contraction and $u \in S$. The new virtual terminal has a weight zero.

Algorithm 3 ***Construct NST***

Repeat the following steps until no receivers are left and there is only one virtual terminal left.

1. Find the spider S with the minimum $\rho(S)$ that connects some receivers and virtual terminals.[2]
2. Contract the spider S by treating all nodes in it as one virtual terminal. We call this as one *round*.

All nodes belong to the final unique virtual terminal form the NST.

The following theorem is proved in [12].

Theorem 7 *[12] Given k receivers, the tree constructed above has cost at most $2 \ln k$ times of the optimal.*

VCG mechanism on NST in not strategy-proof

Again, we may want to pay terminals based on the VCG scheme, i.e., the payment to a terminal $v_k \in NST(d)$ is

$$p_k(d) = \omega(NST(d|^k\infty)) - \omega(NST(d)) + d_k.$$

We show by an example that the payment scheme does *not* satisfy the IR property: it is possible that some terminal has a negative utility under this payment scheme. Figure 1.3

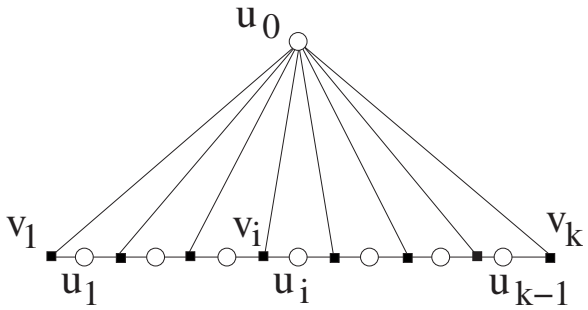


Figure 1.3: Terminals q_i, $1 \le i \le k$ are receivers; the cost of terminal v_{2k-1} is 1. The cost of each terminal v_i is $\frac{2}{k+1-i} - \epsilon$, where ϵ is a sufficiently small positive number.

illustrates such an example. It is not difficulty to show that, in the first round, terminal v_k is selected to connect terminals s and q_1 with cost ratio $\frac{1}{k} - \frac{\epsilon}{2}$ (while all other spiders have cost ratio at least $\frac{1}{k}$). Then terminals s, v_k and q_1 form a virtual terminal. At the beginning of round r, we have a virtual terminal, denoted by V_r formed by terminals v_{k+i-1}, $1 \le i \le r-1$, and receivers q_i, $1 \le i \le r$; all other receivers q_i, $r < i \le k$ are the remaining terminals. It is easy to show that we will select terminal q_{k+r-1} at round r to connect V_r and q_{r+1} with cost $\frac{1}{k+1-r} - \frac{\epsilon}{2}$. Thus, the total cost of the tree $NST(G)$ is $\sum_{i=1}^{k-1}(\frac{2}{k+1-i} - \epsilon) = 2H(k) - 2 - (k-1)\epsilon$.

When terminal v_k is not used, it is easy to see that the final tree $NST(G \backslash u_1)$ will only use the terminal v_{2k-1} to connect all receivers with cost $\frac{1}{k}$ when $\frac{1}{k-1} - \frac{\epsilon}{2} > \frac{1}{k}$. Notice that this condition can be trivially satisfied by letting $\epsilon = \frac{1}{k^2}$. Thus, the utility of terminal v_k is $p_1(d) - c(v_k) = \omega(NST(G \backslash v_k)) - \omega(NST(G)) = -2H(k) + 3 + (k-1)\epsilon$, which is negative when $k \ge 8$, and $\epsilon = 1/k^2$.

Strategyproof Mechanism on Spiders

Notice, the construction of NST tree is by rounds. Following, we show that if terminal v_k is selected as part of the spider with the minimum ratio under a cost profile d in a round i, then v_k is selected before or in round i under a cost profile $d' = d|^k d'_k$ for $d'_k < d_k$. We prove this by contradiction, which assumes that the terminal v_k won't appear before round $i+1$. Notice the graph remains the same for round i after the profile changes, so spider $S_i(d)$ under the cost profile d is still a valid spider under the cost profile d'. Its ratio becomes $\omega_i^k(d) - d_k + d'_k < \omega_i^k(d)$ while all other spiders' ratio keeps the same if they don't contain v_k. Thus, the spider $S_i^k(d)$ has the minimum ratio among all spiders under cost profile d',

which is a contradiction. So for terminal v_k, there exists a real value $B_k^i(d_{-k})$ such that the terminal v_k is selected before or in round i iff $d_k < B_k^i(d_{-k})$. If there are r rounds, we have an increasing sequence

$$B_k^1(d_{-k}) \leq B_k^2(d_{-k}) \leq \cdots \leq B_k^r(d_{-k}) = B_k(d_{-k})$$

Obviously, the terminal v_k is selected in the final multicast tree iff $d_k < B_k(d_{-k})$. Following is our payment scheme based on NST.

Definition 2 *For a node v_k, if v_k is selected in NST, then it gets payment*

$$p_k(d) = B_k(d_{-k}). \tag{1.3}$$

Otherwise, it gets payment 0.

Regarding this payment we have the following theorem:

Theorem 8 *Our payment scheme (1.3) is truthful, and is minimum among all truthful payment schemes for multicast tree based on spider.*

Proof. To prove that it is truthful, we prove that it satisfies IR and IC respectively. Notice that v_k is selected iff $d_k < B_k^i(d_{-k})$, we have $u_k(d) = B_k(d_{-k}) - d_k > 0$, which implies IR. Now we prove that our payment scheme (1.3) satisfies IC by cases. Notice when v_k is selected, its payment doesn't depend on d_k, so we only need to discuss the following two cases:

Case 1: When v_k declares c_k, it is not selected. What happens if it lies its cost upward as d_k to make it not selected? From the IR property, v_k gets positive utility when it reveals its true cost while it gets utility 0 when it lies its cost as d_k. So it better for v_k not to lie.

Case 2: When v_k declares a cost c_k, it is not selected. What happens if it lies its cost downward as d_k to make it selected? When v_k reveals c_k, it has utility 0, after lying it has utility $B_k(d_{-k}) - c_k$. From the assumption that v_k is not selected under cost profile $d|^k c_k$, we have $B_k(d_{-k}) \leq c_k$. Thus, v_k will get non-positive utility if it lies, which ensures v_k revealing its true cost c_k.

So overall, v_k will always choose to reveal its actual cost to maximize its utility (IC property).

Following we prove that our payment is minimal. We prove it by contradiction, suppose that there exists such a payment scheme $\tilde{P}$ such that for a terminal v_k under a cost profile d, the payment to $\tilde{P}_i(d)$ is smaller than our payment. Notice in order to satisfy the IR, the terminal must be selected, so we assume $\tilde{P}_i(d) = B_k(d_{-k}) - \delta$, where δ is a positive real number. Now considering the profile $d' = d|^k(B_k(d_{-k}) - \frac{\delta}{2})$ with v_k's actual cost being $c_k = B_k(d_{-k}) - \frac{\delta}{2}$. Obviously, v_k is selected, from Lemma 1 the payment to v_k is $B_k(d_{-k}) - \delta$. Thus, the utility of v_k becomes $u_k(d') = B_k(d_{-k}) - B_k(d_{-k}) - \delta + \frac{\delta}{2} = -\frac{\delta}{2} < 0$, which violates the IR property. This finished our proof. □

We then study how to compute such payment to a selected node v_k. With Theorem 8, we only need focus our attention on how to find the value $B_k^i(d_{-k})$. Before we present our algorithm to find $B_k^i(d_{-k})$, we first review in details how to find the minimum ratio spider. In order to find the the spider with the minimum ratio, we find the spider centered at every vertex v_j with the minimum ratio over all vertices $v_j \in V$ and choose the minimum among them. The algorithm is as follows.

Algorithm 4 ***Find the minimum ratio spider***

Do the following process for all $v_j \in V$:

1. Calculate the shortest path tree rooted at v_j that span all terminals. We call each shortest path a *branch*. The weight of the branch is defined as the length of the shortest path. Here, the weighted of the shortest path doesn't include the weight of the center node v_j of the spider.
2. Sort the branches according to their weights.
3. For every pair of branches, if they have terminals in common then remove the branch with a larger weight. Assume that the remaining branches are
$$L(v_j) = \{L_1(v_j), L_2(v_j), \cdots, L_r(v_j), \}$$
sorted in an ascending order of their weights.
4. Find the minimum ratio spider with center v_j by linear scanning: the spider is formed by the first t branches such that $\frac{c_j+\sum_{k=1}^{t} L_k}{t} \leq \frac{c_j+\sum_{k=1}^{h} L_k}{h}$ for any $h \neq t$.
Assume the spider with the minimum ratio centered at terminal v_j is $S(v_j)$ and its ratio is $\rho(v_j)$.
5. The spider with minimum ratio for this graph is then $S = \min_{v_j \in V} S(v_j)$.

In algorithm 4, $\omega(L_i(v_j))$ is defined as the sum of the terminals' cost on this branch, and $\Omega_i(L(v_j)) = \sum_{s=1}^{i} \omega(L_s(v_j)) + c_j$. If we remove node v_k, the minimum ratio spider with center v_j is denoted as $S^{-v_k}(v_j)$ and its ratio is denoted as $\rho^{-v_k}(v_j)$. Assume that $L_1^{-v_k}(v_j), L_2^{-v_k}(v_j), \cdots, L_p^{-v_k}(v_j)$ are those branches in ascending order before linear scan.

From now on, we fix d_{-k} and the graph G to study the relationship between the minimum ratio $\rho(v_j)$ of the spider centered at v_j and the cost d_k of a node v_k.

Observation 1 *The number of the legs of the minimum ratio spider decreases over d_k.*

If the minimum ratio spider with the terminal v_k has t legs, then its ratio will be a line with slope of $\frac{1}{t}$. So the ratio-cost function is several line segments. From the observation 1, these line segments have decreasing slopes and thus it has at most r segments, where r is the number of receivers. So given a real value y, we can find the corresponding cost of v_k in time $O(\log r)$ such that the minimum cost ratio spider $S(v_j)$ centered at node v_j has a ratio y. We the present our algorithm to find these line segments as follows.

Algorithm 5 ***Find the ratio-cost function*** $y = \mathcal{R}_{v_j}(x)$ ***over the cost*** x ***of*** v_k

There are two cases here: $j = k$ or $j \neq k$.
Case 1: $j = k$, we apply the following procedures:
Apply steps $1, 2, 3$ of algorithm 4 to get $L(v_k)$.
Set the number of legs to $t = 1$, lower bound $lb = 0$ and upper bound $ub = 0$
While $t < r$ do the follows {
$ub = (t+1) * \omega(L_{t+1}(v_k)) - \Omega_{t+1}(L(v_k))$
$y = \frac{\Omega_t(L(v_k))+x}{t}$ for $x \in [lb, ub)$
Set $lb = ub$ and $t = t + 1$ }
Let $y = \frac{\Omega_r(L(v_k))+x}{r}$ for $x \in [lb, \infty)$.
Case 2: $j \neq k$, we apply the following procedures:

1. Remove terminal v_k, apply algorithm 4 to find $S^{-v_k}(v_j)$.
2. Find the shortest path with terminal v_k from v_j to every receiver, sorted these paths according to their length in a descending order, say sequence
$$L^{v_k}(v_j) = \{L_1^{v_k}(v_j), L_2^{v_k}(v_j), \cdots, L_r^{v_k}(v_j)\}.$$
Here r is the number of terminals, and $\omega(L_i^{v_k}(v_j))$ is the sum of terminals on path $L_i^{v_k}(v_j)$ *excluding* terminal v_k.
3. t is the index for branches in $L^{v_k}(v_j)$ and l is the index for paths in $L^{-v_k}(v_j)$.
4. For $L_t^{v_k}(v_j)$ $(1 \leq t \leq r)$, there may exists on or more branches in $L^{-v_k}(v_j)$ such that they have common terminals with $L_t^{v_k}(v_j)$. If there are more than one such branches, choose the branch with the minimum cost, say $L_l^{-v_k}(v_j)$. We defined upper bound $upper_t$ for $L_t^{v_k}(v_j)$ equals $\omega(L_l^{-v_k}(v_j)) - \omega(L_t^{v_k}(v_j))$. If there does not exist such branch we set $upper_t = \infty$.
5. Initialize lower bound $lb = 0$ and upper bound $ub = 0$. Apply the following algorithm:
For $t = 1$ to r do {
While $lb < upper_t$ do
Set $l = 1$
Obtain a new sequence $LT^{-v_k}(v_j)$ from $L^{-v_k}(v_j)$ by removing all branches that has common nodes with $L_t^{v_k}(v_j)$. Let rt be the number of branches in sequence $LT^{-v_k}(v_j)$.
While $l \leq rt$ do
While $\omega(L_t^{v_k}(v_j) + lb > l\omega(LT_l^{-v_k}(v_j)) - \Omega_{l-1}(LT^{-v_k}(v_j)) - c_j$ and $l \leq rt$
$l = l + 1$
If $l \leq rt$ then
Set $ub = \omega(LT_l^{-v_k}(v_j)) - \Omega_{l-1}(LT^{-v_k}(v_j)) - \omega(L_t^{v_k}(v_j) - c_j$
If $ub \geq upper_t$ break;
Set $y = \frac{\Omega_{l-1}(LT^{-v_k}(v_j))+\omega(LT_t^{v_k}(v_j))+x}{l}$ for $x \in [lb, ub)$
Set $lb = ub$.
Set $l = l + 1$.
Set $y = \frac{\Omega_{l-1}(LT^{-v_k}(v_j))+\omega(L_t^{v_k}(v_j))+x}{l}$ for $x \in [lb, upper_t)$.
Set $lb = upper_t$.
}

Given a real value x, the corresponding cost for terminal v_k is denoted as $\mathcal{R}_{v_j}^{-1}(x)$. Finally, we give the algorithm to find value $B_k(d_{-k})$.

Algorithm 6 ***Algorithm to find*** $B_k(d_{-k})$

1. Remove the terminal v_k and find the multicast tree by using the spider structure.
2. For every round i in the first step, we have a graph called G_i and a selected spider with ratio $\rho_i^{-v_k}$. Adding the node v_k and all its incident edges to G_i, we get a graph G_i'.
3. Find the function $y = \mathcal{R}_{v_j}^{-1}(x)$ for every terminal v_j in the graph G_i' using Algorithm 5.
4. Calculate $B_k^r(d_{-k}) = \max_{v_j \in V(G_i')}\{\mathcal{R}_{v_j}^{-1}(\rho_i^{-v_k})\}$.
5. $B_k(d_{-k}) = \max_{1 \le i \le r} B_k^i(d_{-k})$

The correctness of our algorithms is omitted due to space limit. Notice that for the practical implementations, we do not have to compute the functions actually. We are more interested in given some value y, what is the corresponding cost d_k such that the minimum ratio spider centered at the node v_k has a ratio y.

1.4 Experimental Studies

Remember that the payment of our structure is often larger than the structure's actually cost. For a structure H, let $c(H)$ be its cost and $p_s(H)$ be the payment of a scheme s based on this structure. We define the overpayment ratio of the payment scheme s based on structure H as

$$OR_s(H) = \frac{p_s(H)}{c(H)}. \tag{1.4}$$

When it is clear from the context, we often simplify the notation as $OR(H)$.

Actually, there are some other definitions about overpayment ratio in the literature. In [**?**], the authors propose to compare the payment $p(H)$ with the cost of the new structure obtained from the graph $G - H$, i.e., removing H from the original graph G. Here, we only focus our attention on the overpayment ratio defined as (1.4).

We conducted extensive simulations to study the overpayment ratio of various schemes proposed in this chapter. In our experiments, we will compare the different schemes proposed according to three different metrics: actual cost, total payment and overpayment ratio. Figure 1.4 shows the different multicast structures when the original graph is a unit disk graph. Here, the grey nodes are receivers.

In the first experiment, we randomly generate n nodes uniformly in a $2000ft \times 2000ft$

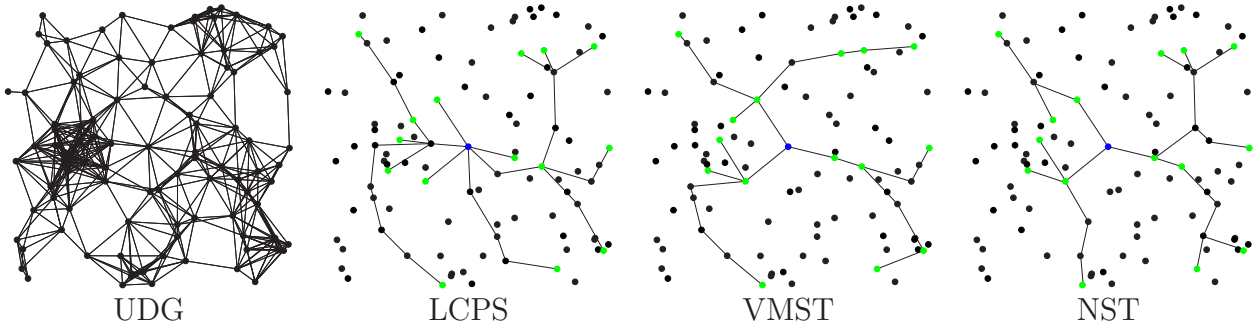
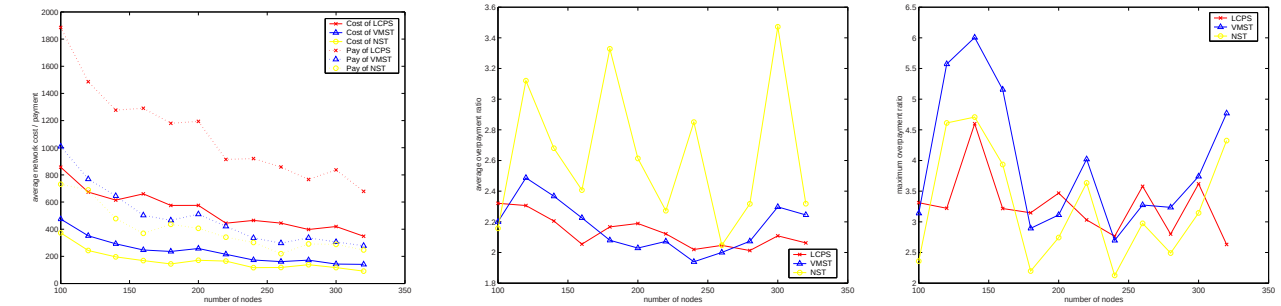


Figure 1.4: Multicast Structures for Node Weighted Network

region. The transmission range of each node is set as $300ft$. The weight of a node i is $c_i * 3^\kappa$ where c_i is randomly selected from a power level between 1 and 10. We vary the number of terminals in this region from 100 to 320, and fix the number of sender to 1 and the number of receivers to 15. For a specific number of terminals, we generate 500 different networks, and compare the average cost, maximum cost, average payment and maximum payment, average overpayment ratio and maximum payment ratio.



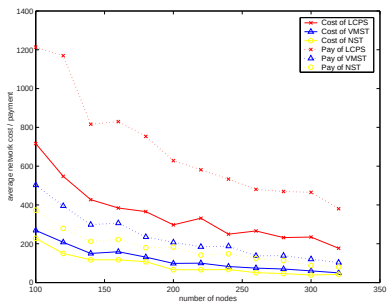
average network cost / payment average overpayment ratio maximum overpayment ratio

Figure 1.5: Results when the number of nodes in the networks are different (from 100 to 320). Here, we fix the transmission range to $300ft$.

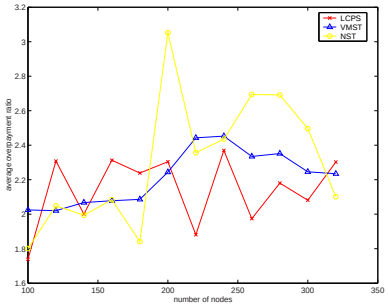
In wireless ad hoc networks with nodes having fixed transmission range, as shown in Figure 1.5, all structures' cost and payment decrease as the number of terminals increase. Notice for all structures, we assume all receivers(senders) will relay the message for free. The cost and payment of VMST and NST are much lower than the cost and payment of LCPS. As we expected, due to the low cost of the VMST and NST structures, the maximum overpayment ratio of these two structures are very unsteady and much high than the max overpayment ratio of LCPS. We still prefer the later structures VMST and NST since they incur smaller costs and more importantly smaller payments.

In our second experiment, we vary the transmission range of each wireless node from $100m$ to $500m$. We assume that the cost c_i of a terminal v_i is $c_1 + c_2 {r_i}^\kappa$, where c_1 takes value from 300 to 500, c_2 takes value from 10 to 50 and r_i is v_i's transmission range. The ranges of c_1 and c_2 we used here reflects the actual power cost in one second of a node to send data at $2Mbps$ rate.

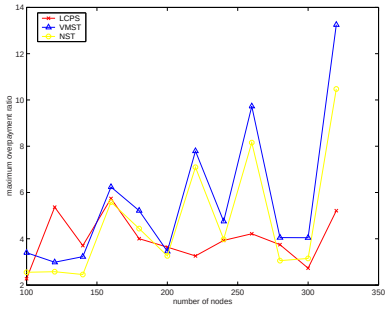
Similar to the fixed transmission experiment, we vary the number of terminals in the region from 100 to 320, and fix number of sender to 1 and the number of receivers to 15. For



average network cost / payment



average overpayment ratio



maximum overpayment ratio

Figure 1.6: Results when the number of nodes in the networks are different (from 100 to 320). Here, we randomly set the transmission range from $100ft$ to $500ft$.

a specific number of terminals, we generate 500 different networks, and compare the average cost, maximum cost, average payment and maximum payment, average overpayment ratio and maximum payment ratio.

Figure 1.6 shows the similar result for both link weighted network and node weighted network as the fixed transmission range experiments.

1.4.1 Vary Number of Receivers and Random Transmission Range

For a structure H, if there are r receivers, we define the cost density as

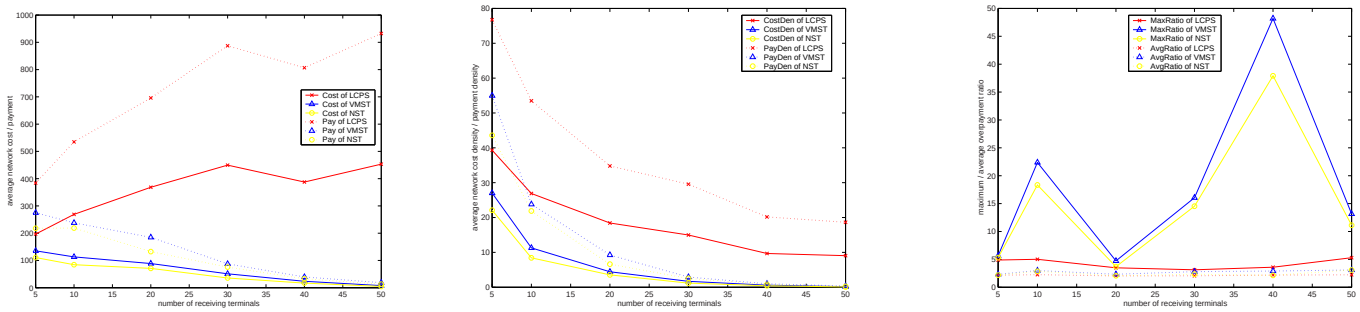
$$CD(H) = \frac{c(H)}{r}$$

and the payment density as

$$PD(H) = \frac{p(H)}{r}.$$

Now we study the relationship between cost, payment, overpayment ratio, cost density, payment density and the number of the terminals. We use the same power cost model in the previous experiment and the number of nodes in the region is set to 300. We fix the sender to 1 and vary the number of receivers from 5, 10, 20, $\cdots$ to 50.

Figure 1.7 shows that when the number of the receivers increases, under most circumstances, the overall payment and cost increases while the average cost and payment for every terminal decreases. One exception is for a node weighted network. Notice that in a node weighted network, we set all terminals' cost to 0. It is natural to expected that when the number of receivers is larger than some threshold, the total cost and payment will decrease since we assume that receivers will relay for free. This experiment shows that more terminals in a multicast group can incur a lower cost and payment per terminal, which is an attractive property of multicast.



average network cost / payment average cost / payment density average /maximum overpayment ratio

Figure 1.7: Results when the number of receivers in the networks are different (from 10 to 50). We randomly set the transmission range from $100ft$ to $500ft$.

1.5 Conclusion

In this chapter, we studied how to conduct efficient multicast in *selfish* wireless networks by assuming that each wireless node will incur a cost when it has to transit some data, and the cost is privately known to the wireless terminal node. For each of the widely used structures for multicast, we designed a strategyproof multicast mechanism such that each agent maximizes its profit when it truthfully reports its cost. The structures studied in this chapter are: least cost path star(LCPS), virtual minimum spanning tree(VMST) and the node weighted Steiner tree(NST). We showed that the VMST approximates the optimal multicast tree for the homogeneous network when all wireless nodes have the same transmission range.

Extensive simulations were conducted to study the practical performances of the proposed protocols, especially the total payment of these protocols compared with the actual cost of agents. Although, theoretically, the overpayment ratio of the protocols designed here could be large in the worst scenario, we found by experiments that the overpayment ratios of all our strategyproof mechanisms are small when the costs of agents are randomly drawn from a distribution.

References

[1] Anderegg, L., and Eidenbenz, S. Ad hoc-vcg: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *Proceedings of the 9th annual international conference on Mobile computing and networking* (2003), ACM Press, pp. 245–259.

[2] Blazevic, L., Buttyan, L., Capkun, S., Giordano, S., Hubaux, J. P., and Boudec, J. Y. L. Self-organization in mobile ad-hoc networks: the approach of terminodes. *IEEE Communications Magazine 39*, 6 (June 2001).

[3] Buttyan, L., and Hubaux, J. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications 5*, 8 (October 2003).

[4] Buttyan, L., and Hubaux, J. P. Enforcing service availability in mobile ad-hoc wans. In *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing* (2000), pp. 87–96.

[5] Clarke, E. H. Multipart pricing of public goods. *Public Choice* (1971), 17–33.
[6] Feigenbaum, J., Papadimitriou, C., Sami, R., and Shenker, S. A BGP-based mechanism for lowest-cost routing. In *Proceedings of the 2002 ACM Symposium on Principles of Distributed Computing.* (2002), pp. 173–182.
[7] Feigenbaum, J., Papadimitriou, C. H., and Shenker, S. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences 63*, 1 (2001), 21–41.
[8] Green, J., and Laffont, J. J. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica* (1977), 427–438.
[9] Groves, T. Incentives in teams. *Econometrica* (1973), 617–631.
[10] Guha, S., and Khuller, S. Improved methods for approximating node weighted steiner trees and connected dominating sets. In *Foundations of Software Technology and Theoretical Computer Science* (1998), pp. 54–65.
[11] Jakobsson, M., Hubaux, J.-P., and Buttyan, L. A micro-payment scheme encouraging collaboration in multi-hop cellular networks. In *Proceedings of Financial Cryptography* (2003).
[12] Klein, P., and Ravi, R. A nearly best-possible approximation algorithm for node-weighted steiner trees. Tech. Rep. CS-92-54, 1992.
[13] Marti, S., Giuli, T. J., Lai, K., and Baker, M. Mitigating routing misbe-havior in mobile ad hoc networks. In *Proc. of MobiCom* (2000).
[14] Nisan, N., and Ronen, A. Algorithmic mechanism design. In *Proc. 31st Annual Symposium on Theory of Computing (STOC99)* (1999), pp. 129–140.
[15] Salem, N. B., Buttyan, L., Hubaux, J.-P., and Kakobsson, M. A charging and rewarding scheme for packet forwarding in multi-hop cellular networks. In *ACM MobiHoc* (2003).
[16] Srinivasan, V., Nuggehalli, P., Chiasserini, C. F., and Rao, R. R. Energy efficiency of ad hoc wireless networks with selfish users. In *European Wireless Conference 2002 (EW2002)* (2002).
[17] Srinivasan, V., Nuggehalli, P., Chiasserini, C. F., and Rao, R. R. Cooperation in wireless ad hoc wireless networks. In *IEEE Infocom* (2003).
[18] Vickrey, W. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance* (1961), 8–37.